Trusted Relay Networks

A Mechanism for Easy Scalability of the Ethereum Network

Version 1.0
By: Alex Miller

Motivation

One of Ethereum's most widely discussed shortcomings is its lack of scalability - a problem that persists for all public blockchains. All currently implemented distributed blockchain networks, including Ethereum, require every node in the network to process every transaction. Security comes through redundancy of computation (ideally across decentralized control of nodes) at the expense of transactive throughput. Several scaling solutions have been proposed, but most remain in the theoretical stage. Some of the most promising solutions include Raiden[1] (state channel networks), state sharding[2], and hierarchical plasma chains[3]; implementing any of these solutions represents a significant technical challenge and each has a long roadmap before any are deemed production-ready. Meanwhile, usage of Ethereum continues to increase - with daily transactions routinely nearing one million.

Surprisingly, it is seldom discussed that external blockchains which utilize the Ethereum Virtual Machine (EVM) could represent additional (linear) capacity for the Ethereum network. For example, Ethereum Classic uses roughly 200,000 gas per block[4] which is more than 30x lower than Ethereum's gas usage of roughly 6.7m per block[5]. However, applications cannot currently cross networks, which means the vast majority of assets and functionality is constrained to the main Ethereum network (ETH). Although functionality will always be blockchain specific (due to the constraints of deterministic systems), assets can theoretically be moved between chains so long as only one representation is being used at any point in time (with all other versions locked). If users had the ability to seamlessly move assets between multiple networks, they would likely shift transactional throughput to the less constrained blockchain in times of high load and return to the busier chain when liquidity is desired. Unfortunately, there is currently no production method to move assets between any two blockchains at scale.

Atomic swaps[6] are a method for *trading* assets between accounts on a single chain or across chains. Several implementations exist, including a simple Ethereum-based contract created by the author as an experiment.[7] In this Ethereum-based design, Alice and Bob both put their assets in a "locked box" on their respective blockchains, which requires both signatures to open. Either Alice or Bob passes a signature to the other, who uses it to unlock the desired box. Once unlocked, the box emits *both* signatures and the counterparty uses those to unlock his or her box. The net result is that Alice and Bob have traded assets *between blockchains*.

However, atomic swaps do not solve the problem of temporarily or permanently *moving* a user's assets - the assets swapped remain on their respective chains and it is the users themselves that cross networks. Because a user is represented by a private/public key pair in every blockchain-based system, Alice and Bob simultaneously exist as actors on both networks.

This paper proposes a novel scheme to move assets between EVM-based blockchains in a way that requires minimal trust in a relayer, who is incentivized to relay a message to the desired network. This requires the asset be locked in the initial chain and copied onto the new blockchain before any such movement can occur. Because this scheme is based on an Ethereum smart contract, the "user" is represented by a single address corresponding to a key pair which can be used on any number of EVM-based blockchains. This single address can represent a single user, or the public address of a multi-signature contract. The totality of this system is termed a *Trusted Relay Network*.

Relays

A relay is a mechanism to move tokens and other assets between EVM-based blockchains. This design requires a trusted relayer (or set of trusted relayers) to deliver a message from one blockchain to another.

Relayers operate around multiple associated "Gateway" contracts, which exist on both the *origin* and *destination* blockchains. Each Gateway has all trusted relayers encoded as *owners* of the contract, where only an owner can relay a message from the Gateway in the origin chain to the Gateway in the destination chain.

Figure 1. A simple Trusted Relayer Network with two chains and one relayer

A relay network is the set of smart contracts on each of these networks. The contracts are "owned" by a relayer, who is represented by a private/public key pair, which may either be a single account or a multi-sig contract deployed to each of the respective chains. A simple Trusted Relayer Network is shown in Figure 1, where the origin is "Chain 1" and the destination is "Chain 2".

Procedure

The relay procedure happens in two steps.

Step 1: User calls function on the origin chain

The user calls a function of the Gateway contract on the origin chain with the following information:

a. `<chainId>` The address of the Gateway contract on the destination chain. This serves as a de-facto identifier of the chain itself within the scope of the Trusted Relayer Network.
b. `<token>` Address of the token deposited on the origin chain.
c. `<amount>` Amount of token (in atomic units) deposited on the origin chain.
d. `<fee>` Fee to be sent to the relayer (in the withdrawn token on the destination chain). This can be zero.
e. `<timestamp>` Time at which this message was signed. This is mostly used as a nonce.
f. `<hash>` Hash of the above information (see below).
g. `<signature>` v, r, s values of the elliptic curve signature of the hash.

Where the hash includes the following:

```
keccak256(<prefix>, <origin chainId>, <destination chainId>, <origin token address>, <user address>, <amount of token deposited (atomic units)>, <fee>, <timestamp>)
```

With the prefix corresponding to an Ethereum signed message standard outlined in ERC191.[8]

Upon calling the deposit function, the Gateway smart contract verifies the hash and signature are correct given the provided data and transfers tokens[1] from the user to itself. This requires the user to approve a transfer ahead of time. The tokens are now locked in the Gateway contract on the origin chain.

Step 2: Relayer sends message to destination chain

[1] A token is any asset that can be moved on behalf of the owner and replicated on a separate blockchain. The prototypical example is the ERC20 specification[9], which is a representation of fungible tokens. Non-fungible tokens may also be relayed (e.g. ERC721 tokens[10]) so long as they have the requisite `transferFrom` functionality.

The relayer takes the message from above (along with the corresponding data) and calls a function on the destination chain. This finds the token associated with the token address specified.

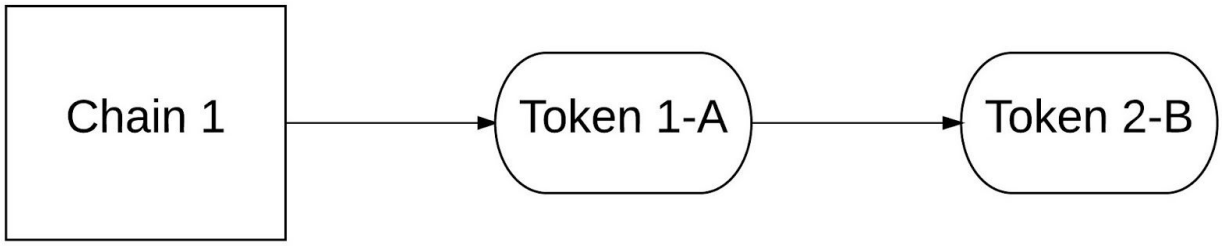

Figure 2: An Example Relay Lookup in Chain 2

This is an object stored in the Gateway smart contract state mapping a destination chainId and origin token address to a destination token address. Figure 2 shows an example mapping, which would exist in the Gateway contract of Chain 2. This mapping shows that Token 1-A on Chain 1 maps to Token 2-B on Chain 2.

Need for Trust

Blockchains are deterministic systems, which means it is impossible to *prove* an event (e.g. locking tokens) has occurred from within a smart contract running on-chain without outside input. Although there are several ways to circumvent this obstacle, Trusted Relayer Networks take a simple approach by only allowing trusted relayers to trigger token distributions. A relayer waits for a user to lock tokens, which emits an event. Once the event is emitted from the origin chain, the relayer takes the message to the destination chain and triggers another event, which unlocks tokens for the original user. For this service, the relayer may charge a fee. If a user were left in charge of this functionality, he or she could simply recreate the message without locking up the origin tokens and double his or her token balance.

Fees

The fee included in the original message is taken from the withdrawn tokens (i.e. in the destination chain) and sent to the `msg.sender` (i.e. the relayer). This means that no fee is paid up-front by the sender to the relayer and is only realized on the destination chain. Normal gas costs still apply for deposits.

As with blockchain systems such as Bitcoin and Ethereum, the user determines the fee. If the relayer is insufficiently incentivized, the user risks not having his or token transactions relayed. However, a relayer is also incentivized to run a reputable operation. Thus, a relay network

would likely publish a minimum fee to a user who wants to use their service. In many cases, this fee would be zero (see: Relayer Incentives).

Asset Replication

Although relays may seem similar to atomic swaps, they differ in one fundamental regard: relays move assets between chains with *no counterparties*. An atomic swap is a trade between two actors, while a relay transmits a user's message from one blockchain to another. This means that any assets being relayed must first be *replicated* in the destination chain.

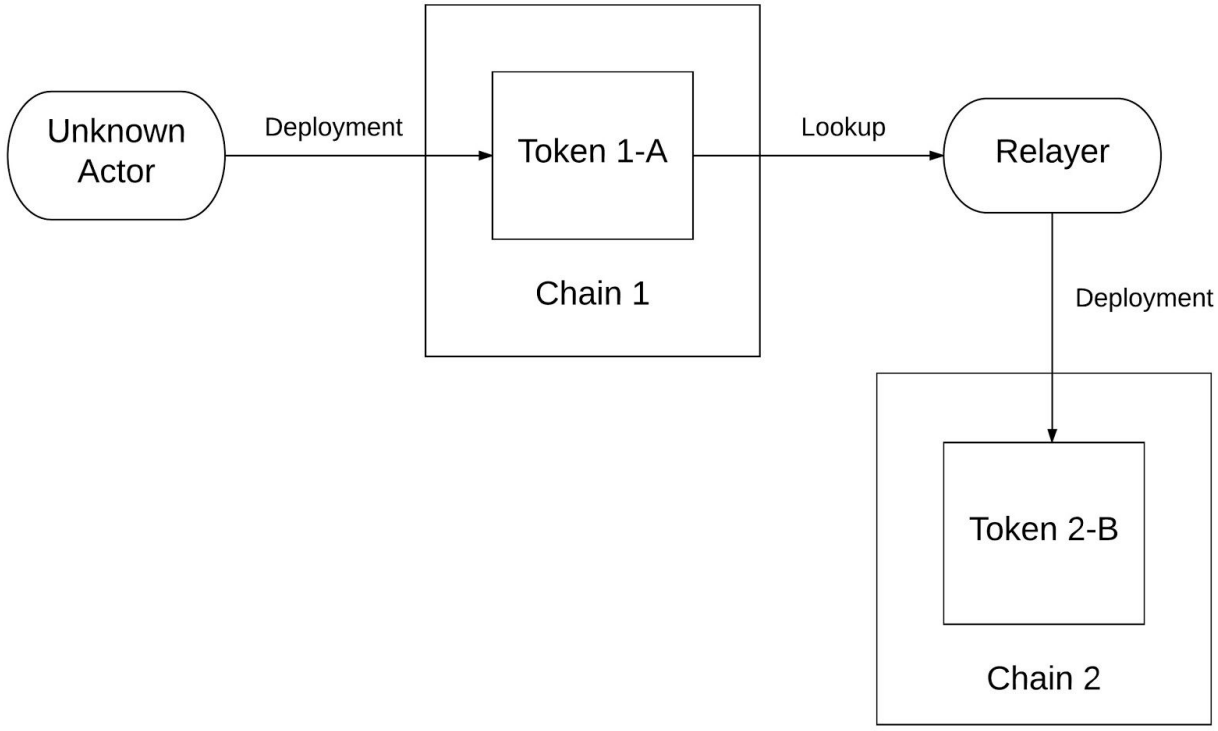

Figure 3: Asset Replication

Creating New Tokens

Before any message can be relayed, a mapping from the origin token to the destination token must exist. However, a mapping to a non-existent token cannot be made, so the token must first be replicated onto the destination chain.

Replication is done by any relayer (who is an owner of the destination Gateway) and involves creating a transaction with the origin token's bytecode and its metadata (name, decimals, symbol, supply, etc). Figure 3 shows this replication process. An unknown actor deploys Token 1-A to Chain 1. The relayer receives a message (not pictured) from the Gateway in Chain 1 and looks up Token 1-A's bytecode before re-deploying it to Chain 2. Token 2-B is deployed and is associated with Token 1-A and Chain 1 on the Gateway contract in Chain 2 (see: Figure 2).

This highlights an important invariant in the system: *all deposits must map 1:1 across chains*. If 100M Token 1-A is deposited to Chain 1, then 100M Token 2-B must be deposited by an owner to the Gateway in Chain 2. Thus, deposits to a new (i.e. unreplicated) asset are dangerous. However, once the replicated asset is deposited and mapped (a mapping cannot be overwritten), the user does have assurance that there will be no arbitrary inflation of the token on the destination chain (relative to the origin chain).

Edge Case Protection

Since the raw bytecode must be replayed, not all tokens will be possible to replicate. Certain tokens might have unpredictable issuance patterns or instantiation functions that require some logic the relayer is not capable of performing (e.g. signatures to prove data). In the event that a token is not replicable, its mapping will be marked with `address(1)` (or simply flagged in an internal database) and any pending relay requests will be cancelled (see: Cancellations).

Ether Tokens

While most relays will move ERC20 tokens into new, replicated ERC20 tokens, it is also possible to mark one token per origin chain[2] as an `etherToken`. This token, once deposited into the origin Gateway and relayed, will convert[3] to ether on the destination chain.

If an ether token is specified, a separate mapping exists between a certain number of ether tokens and either a fixed or growing supply of ether. If the supply is fixed, the token to ether mapping would likely be 1:1. If the ether supply grows (via e.g. proof-of-work mining), the ether tokens might function as a “pre-mine” on the origin chain. It is important to consider that not all ether liquidity could be pulled out of the destination chain in the latter scenario.

Returning to the Origin Chain

Because Gateway contracts exist in both the origin and destination chain, the process to return assets to the origin chain mirrors the initial process to bring them to the destination chain. On the origin side, the mapping would associate the destination token to the origin token[4].

---

[2] Note again that bookkeeping is important! Some subset of the ether (or the entire supply, in the case of a fixed-cap chain) must map 1:1 to all tokens. This can be divided between several ether tokens on multiple chains, but the mappings cannot overlap! For example, if there is 100M ether on Chain 2, having 50M ether tokens on Chain 1 and 50M ether tokens on Chain 3 is okay, but 100M ether tokens on Chain 1 and 100M ether tokens on Chain 3 is **not** okay.

[3] Ether tokens need not map 1:1 to ether, as there is an optional “multiplier” parameter. If, for example, an ether token has 10 decimal places, the multiplier would be 10^8 because ether has 18 decimals. It is important to note that ether tokens must be smaller or equal in number to the mapped ether (i.e. there cannot be more ether tokens than ether).

[4] If an ether token exists on an origin chain, it maps to the zero address, which is meant to represent ether.

As an example, suppose the origin chain (chainId=1) holds TKN 1-A and the destination chain (chainId=2) holds TKN 2-B. One mapping would exist on each chain and would be a mirror image of the other (Figure 4).

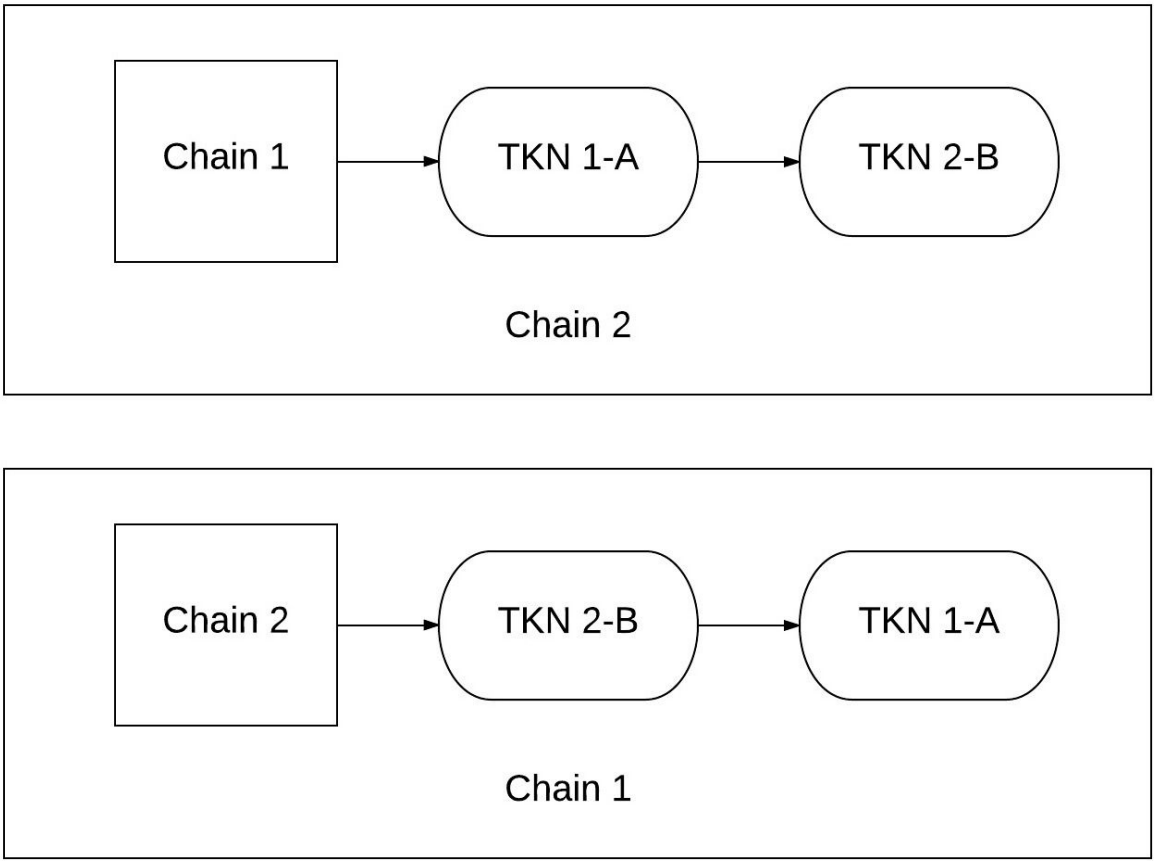

Figure 4: Symmetric mappings of associated cross-chain assets

The same relayer would be an owner of both Gateway contracts and may relay a withdrawal request from the destination chain to the origin chain. Because the origin Gateway still has at least original number of tokens under deposit, it would be able to release tokens to the new owner.

Cancellations

A cancellation can be used to unwind a deposit made in a Gateway and refund the user's tokens on the origin chain. If tokens have been issued on the destination chain, cancellations should only be used as a last resort and once the destination chain has been abandoned. This is a dire action because if the message has already been relayed, cancelling it would break the global invariant which ensures a 1:1 mapping for all associated assets.

Cancellations may be used more commonly if a relayer is insufficiently incentivized to relay the message and has no intention of relaying the message in the future. A cancellation function is called with the original data and the user's signature to unwind the deposit. Unlike the deposit, this function also requires the *relayer's* signature on the original data. Thus, if a relayer is unwilling to spend gas on the destination chain (i.e. believes the fee is too low), he or she may sign a message (for free) and give it to the user to unwind the deposit.

Bootstrapping

Why would a relay network emerge in the first place? Suppose the creators of an application that requires many transactions want to offload functionality onto their own network. By connecting to the ETH mainnet, users could move value into and out of the new app-chain seamlessly. While in the app-chain, users would enjoy vastly diminished transaction fees.

As a relay chain (or any non-mainnet chain) acquires popularity, relayers may become incentivized to relay messages to and from this network. If there is sufficient liquidity in the tokens being moved, one might expect third party relayers to emerge and further ratify the connectivity to the mainnet.

Relay chains are especially suited for applications that require many transactions, low interaction with 3rd party contracts, and relatively few assets. While technically possible to replicate any dependencies, the complexity increases for any applications that are not self-contained. Thus, applications that do not strongly benefit from being on the Ethereum mainnet might be more likely to co-exist on one or more relay chains.

Limitations

There are shortcomings to Trusted Relay Networks, with the most important being the relative ease with which the global invariant (1:1 mapping) may be broken. Users must trust that relayers will not break the invariant in the future, believing that a) the relayers are sufficiently incentivized to continue their business of relaying, b) there is insufficient reward in cancelling deposits made by users.

Reputation

Although outside the scope of this scheme, adding reputation to any Trusted Relayer Network would be a significant improvement. Because all function calls emit events, the global invariant can be validated by anyone with a connection to both relay chains.

It is conceivable that auditing bodies may emerge for relayers - these independent auditors might scan events corresponding to relayed messages and provide a stamp of approval indicating good behavior. One could imagine a scenario where these auditors are paid a small

subscription fee by a userbase that, among other things, needs to utilize Trusted Relayer Networks in daily life.

Conclusion

Relay networks bring additional “virtual” capacity to the Ethereum network by replicating assets on external blockchains. They can be used today for inexpensive and low-risk scaling; inexpensive because new destination chains will almost always have low usage and therefore inexpensive transaction fees and low-risk because any deposits can be rolled back in the case of catastrophic failure.

Although this scheme is titled “Trusted” Relay, the level of trust is minimized relative to a federated peg.[10] The relayer is unable to steal assets and is incentivized through a fee paid after the message is successfully relayed. Furthermore, the user’s signature is required to relay the message, which can only be relayed once. Finally, all deposits and replays trigger events that are publicly viewable by anyone connected to the origin and destination chains.

Thus, the degree of trust is minimized to the relayer not breaking global 1:1 asset mapping invariants and actually relaying messages that have been requested. If the relayer engages in verifiably malicious behavior or stops relaying messages in a timely fashion, one can reasonably expected reduced usership of the relay system.

Although not a perfect solution, the simplicity of the Trusted Relay Network makes it a solution ready to ameliorate today’s scaling problems and a bootstrapping mechanism to drive traffic to a new chain that was created by an actor who wants to promote its use.

References

[1] https://raiden.network/
[2] https://github.com/ethereum/sharding
[3] http://plasma.io
[4] http://gastracker.io/
[5] https://etherscan.io/blocks
[6] https://bitcointechtalk.com/atomic-swaps-d6ca26b680fe
[7] https://github.com/alex-miller-0/eth-dev-101/blob/master/truffle/contracts/AtomicSwap.sol
[8] https://github.com/ethereum/EIPs/issues/191
[9] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
[10] https://github.com/ethereum/EIPs/issues/721
[11] https://blockstream.com/strong-federations.pdf